Research Paper II

Cloud Software Testing

SE 317

Khushveen Kaur Umra

Section 2

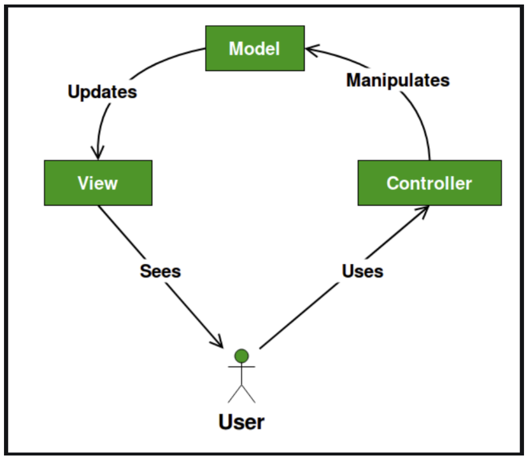
Iowa State University

5th July 2022

Research Paper II – Cloud Software Testing

Part I: Introduction to the MVC Design Pattern:

The Model-View-Controller (MVC) design pattern specifies that an application consists of a data model, presentation information, and control information. This refers to the three primary elements the framework uses to save, retrieve, manipulate, and display information to the users. The pattern requires that each of these be separated into different objects, as it serves the purpose of separating the presentation layer for the business logic. MVC is more of an architectural pattern, but not for the complete application. This design pattern is used to decouple user interface, data, and application logic. It is also a predictable software design pattern that can be used across many frameworks with many programming languages, commonly Python, Ruby, PHP, JavaScript, and more. It is popularly used to design web applications, and mobile apps, and for developing GUIs that have these visualizations for the user.



Design Components:

- **Model**: The **Model** represents an object or JAVA POJO carrying data. It defines what data the app should contain. If the state of this data changes, then the model will usually notify the view and sometimes the controller. It is independent of the user interface and controls the logic and rules of the application.
- **View:** The **View** represents the visualization of the data that the model contains, to allow the user to see these visualizations. It contains all the functionality that directly interacts with the user - like clicking a button or an enter event.
- **Controller:** The **Controller** acts on both the model and view. It controls the data flow into a model object and updates the view whenever data changes. The controller receives input from the view, then it uses logic to translate the input to a demand for the model, then the model grabs the data, and the controller then passes on the data from the model back to the view for the user to see in a nice display.

MVC in Web Applications:

The MVC design pattern helps in the reusing of code and parallel development. The components created through the MVC design pattern are independent of each other in nature. This specific feature helps the developers to reuse the components and codes easily and quickly in other multiple applications. Although originally, the MCV design pattern was developed for desktop computing, it has now been widely adopted as a design for World Wide Web applications in major programming languages. Several web frameworks have been created that enforce the pattern. Some web MVC frameworks take a thin client approach that places almost the entire model, view, and controller logic on the server. In this approach, the client sends either hyperlink requests or form submissions to the controller and then receives a complete and updated web page from the view: the model exists entirely on the server. These software frameworks vary in their interpretations, mainly in the way that the MVC responsibilities are divided. It is much easier to design a web application architecture using the MVC pattern, as it offers full control over the HTML as well as the URLs. It further makes implementing adaptive rendering techniques easier by helping developers design UI using HTML and JavaScript frameworks that provide consistently high-quality cross-platform browning experiences. MVC also makes test-driven development easier. This can reduce the amount of code and time needed to complete an application. Developers who choose to use MVC, have the opportunity to create one application that functions on multiple devices and platforms, which drastically reduces the total cost of application development. (https://www.educba.com/what-is-mvc-design-pattern/)

Benefits of using MVC:

MVC architecture can be considered a one-of-a-kind approach to building a web application, as the MVC frameworks come packed with the following advantages, making it easier to implement them into your framework.

1) **Separation of concerns:** Each component has its own responsibility, such as differentiating between modules handling input (Controller) versus producing output to users (View).
2) **Extensibility:** New View or Controller objects can be added when a new interface medium is introduced. The model also allows new functionalities to be added independently of other components. Hence, MVC architecture allows changes to be made easily.
3) **Accelerated Development Process:** Whenever an MVC architecture is being employed in developing a web application, one developer can concentrate on the view component whereas the other can focus on the controller and create business logic. Hence in comparison, the MVC design pattern results in higher development speeds, i.e., up to three times faster than the other development models available to the users.
4) **Ideal for Developing Large Size Web Applications:** MVC architecture works exceptionally well when the web application demands the support of a huge developer team, and for web designers who require complete control over the application's behavior.

Implementing the MVC architecture can result in lesser expense of time and money, and the ability to create multiple views makes it the best possible architecture for web application development.

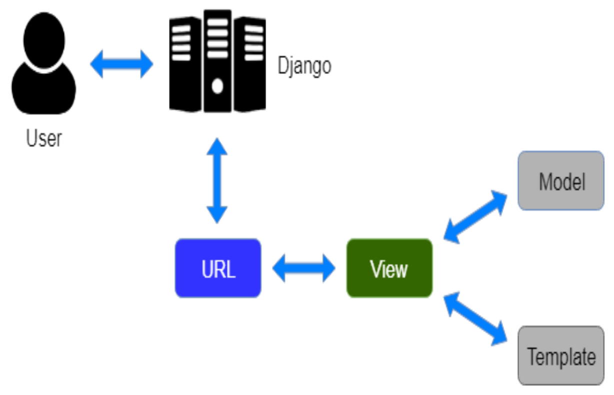
MVC Variants in Programming Languages:

MVC was originally conceptualized in 1976 as a development architecture for creating desktop applications. It has since evolved into a framework for creating crop-platform compatible applications using a variety of programming languages. The MVC pattern is widely used in program development with programming languages such as Java, Smalltalk, C, and C++. The MVC concept is very helpful as it gives you an outline of how the ideas should be organized into actual code. In object-oriented programming development, model-view-controller (MVC) is the name of a methodology or design pattern for successfully and efficiently relating the user interface to underlying data models.

In the Java Programming context, the Model consists of simple Java classes, the View displays the data and the controller consists of servlets. This separation results in user requests being processed as follows:

- The browser on the client-side sends a request for a page to the controller present on the server.
- The controller performs the action of invoking the model, thereby, retrieving the data it needs in response to the request. The controller then gives the retrieved data to the view.
- The view is rendered and sent back to the client for the browser to display.

Another variant example of MVC can be seen in the use of Django. Django is a website framework made with Python. Specifically, the architecture adopted by Django is named Model-View-Template (MVT), which has slight differences from MVC. Here, the web MVC usually includes additional responsibility for the Controller component - to handle the initial HTTP request. Whereas, using the MVT architecture, Django encourages rapid development with a clean design. It also enhances security and allows applications to scale. As Django works on the concept of MVT, it itself takes care of the controller part and the user does not have to worry about how the data moves between the model and the view. Using the Django Template Language, one template can be used by multiple views to show different kinds of data. The templates (or HTML pages) contain the data output response from the views and are then rendered by the browser at the specified URL.



The MVT model functions as follows:

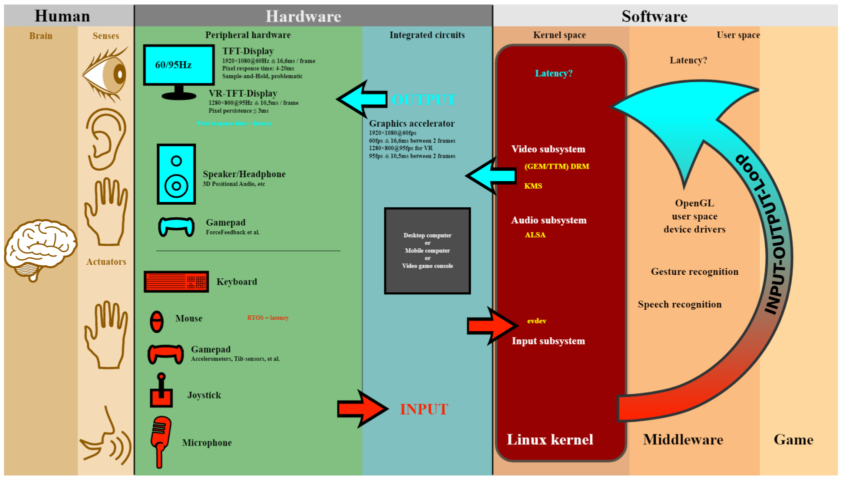
- The Model is the middleware that handles data between the database and the view using the specified definitions for the data fields.
- The view receives the data as well as a request from the client and then accordingly processes the data in the database via the models.
- This data is then available on the user interface with the help of the templates.

Like in Django, MVC can be further modified to meet different requirements and product designs. This way, we have another variant known as the Model-View-ViewModel (MVVM), which can be seen in the C# programming language. It is a software design pattern that is structured to separate a program logic and user interface controls. MVVM was designed to remove virtually all GUI code (“code-behind”) from the view layer, by using the data binding function in WPF (Windows Presentation Foundation) to better facilitate the separation of view layer development from the rest. In MVVM, all communication between Model and View must occur through ViewModel. Here the view is lightweight and a simple component with reduced responsibilities, that only contains UI-specific elements. The controller component in MVVM is named ViewModel with tighter coupling between Model and View. Hence it encapsulates presentation logic and data for the view. ViewModel contains the state of the view and uses Commands, Data binding, and notifications to communicate with the view.
(https://medium.com/@yiyoupteltd/model-view-controller-and-its-variants-2b40403c740f)

Part 2: Description of Concepts:

1) **User Interface Design:**

User Interface (UI) Design focuses on anticipating what users might need to do and ensuring that the interface has elements that are easy to access, understand, and use to facilitate those actions. It is the process designers use to build interfaces in software or computerized devices, focusing on looks or style. The word "interface" refers to the access point where a user interacts with a software application, a browser-based website, or a hardware device. Further, UI Testing, also known as GUI Testing, is basically a mechanism meant to test the aspects of any software that a user will come into contact with. This usually means testing the visual elements to verify that they are functioning according to the requirements. UI testing ensures that all the implemented UI functions are bug-free and ready for the user. UI tests can be conducted manually or they can be automated. In manual testing, a tester manually uses all the features of the website or the application to check for any discrepancies, which makes it inefficient, time-consuming, and prone to human error. In automated testing, automated selenium testing allows the software to be put through multiple test scenarios, and for the same to be run repeatedly quickly, and correctly, which makes them not prone to human error and exhaustion. (https://maze.co/collections/ux-ui-design/ui-design/)

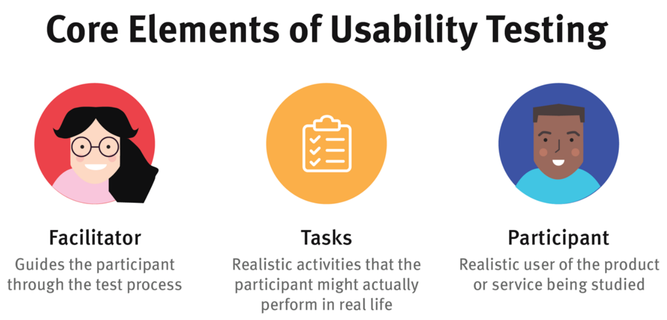


2) **Interaction Design:**

Interaction Design (IxD) is the practice of designing interactive digital products and services. According to John Kolko, the author of 'Thoughts on Interaction Design', interaction design is the creation of a dialogue between a person and a product, system, or service, and this dialogue is both physical and emotional in nature and is manifested in the interplay between form, function, and technology as experienced over time. The ultimate goal of interaction design is to create a solution that fits well in the context of use. To test an interaction design, a tester uses usability testing, which can help them to find design flaws. In a usability testing session, a researcher asks a participant to perform tasks, usually one or more specific user interfaces. While the participant completes each task, the researcher observes the participant's behavior and listens for feedback.

3) **Usability Design and Usability Testing:**

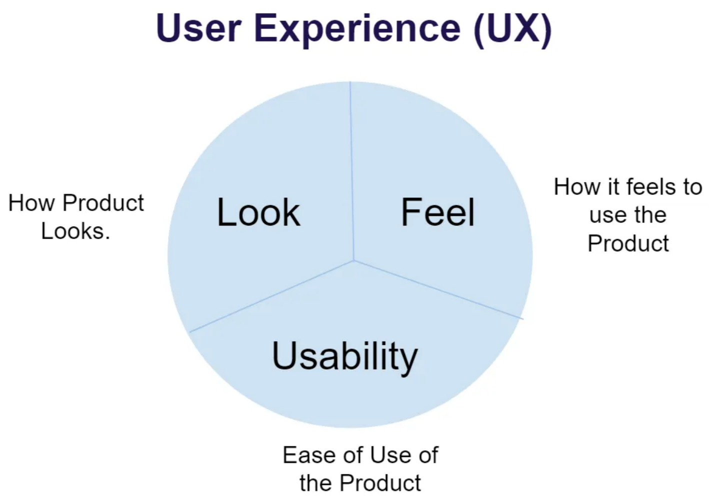
Usability is a measure of how well a specific user in a specific context can use a product/design to achieve a defined goal effectively, efficiently, and satisfactorily. Designers usually measure a design's usability throughout the development process - from wireframes to the final deliverable, to further ensure maximum usability. Usability is the outcome of a user-centered design process. That is the process that examines how and why a user will adopt a product and seeks to evaluate the use. That process is an iterative one and seeks to continuously improve following each evaluation cycle. Whereas, usability testing is all about getting real people to interact with a website, an app, or other products and observing their behaviors and reactions to it. The users are usually observed by researchers working for a business.


Core Elements of Usability Testing
Facilitator
Guides the participant through the test process
Tasks
Realistic activities that the participant might actually perform in real life
Participant
Realistic user of the product or service being studied

The goal of usability testing is to reveal areas of confusion and uncover opportunities to improve the overall user experience. There are two types of usability testing, qualitative testing, where they focus on collecting insights, findings, and anecdotes about how people use the product or service, and quantitative testing, where they focus on collecting metrics that describe the user experience. Usability testing is done by real-life users, who are likely to reveal issues that people familiar with a product or service can no longer identify.

4) <u>User Experience:</u>

The general definition of user experience is the overall experience of a person using a product such as a website or a computer application, especially in terms of how easy or pleasing it is to use it. It is a broad term that can cover anything from how well the user can navigate the product, how easy it is to use, and how relevant the content displayed is. In the digital design world, user experience (UX) refers to everything that affects a user's interaction with a digital product. User experience also depends on the context in which the product is used, as it is essential to understand the role that a product plays in users' lives. Although, a user's experience may change over time.



Part 3: References:

1) https://www.browserstack.com/guide/ui-testing-guide#:~:text=UI%20Test%20Scenarios-,What%20is%20UI%20Testing%3F,terms%20of%20functionality%20and%20performance.
2) https://maze.co/collections/ux-ui-design/ui-design/
3) https://www.onlinetutorialspoint.com/django/django-model-view-template-mvt-overview.html
4) https://www.guru99.com/mvc-tutorial.html
5) https://medium.com/@yiyoupteltd/model-view-controller-and-its-variants-2b40403c740f
6) https://medium.com/edureka/mvc-architecture-in-java-a85952ae2684
7) https://www.educba.com/what-is-mvc-design-pattern/
8) https://www.hotjar.com/usability-testing/
9) https://www.interaction-design.org/literature/article/an-introduction-to-usability
10) https://www.nngroup.com/articles/usability-testing-101/